MISE EN OEUVRE DE LA QUALITE DE SERVICES SOUS LINUX :

CONTROLE ET REPARTITION DU TRAFIC RESEAU

F. DOUCHANT, mai 2005
fabrice.douchant@laposte.net

SOMMAIRE

1) Présentation EOLE Page 3

2) Introduction à la Qualité de Services (QoS) : Page 4

a. Pourquoi la QoS ?
b. Caractéristiques de la QoS Page 5
c. Problématique Page 6
d. La QoS sous Linux Page 7

3) Etude de l'existant : Page 9

a. Rapport de F. Pastor sur le script HTB
b. Comparatifs de scripts QoS
c. Problème et solution

4) Déroulement du projet : Page 10

a. Gestion de 2 interfaces
b. Gestion de 3 interfaces Page 12
c. Gestion de 3 interfaces + Proxy + priorité des flux Page 15

5) Conclusion Page 18

1) Présentation EOLE

Le projet EOLE (Ensemble Ouvert Libre et Evolutif) :

Le projet EOLE est financé par le Ministère de l'Education Nationale. Son objectif est de proposer **des solutions** pour la mise en place de réseaux informatiques pour les **établissements** (lycées et collèges). Pour ses développements, EOLE utilise des logiciels libres, gratuits, soumis à la **licence GPL** (Licence Publique Générale) et adaptables à ses besoins.

L'équipe de projet EOLE s'articule en trois pôles :

- Intégration, veille technologique
- Développement
- Validation, diffusion des modules

Elle compte aujourd'hui 9 membres (dirigés par Luc Bourdot) et accueille tout au long de l'année plusieurs stagiaires pour le développement de projets annexes (ex : Clients Légers etc.) comme le projet qui m'a été assigné : la **Qualité de Services** sous Linux. Mon projet s'est déroulé du 24 février au 18 mars 2005 et a pour but d'être mis en place sur Amon 2 *(1)*.

(1) AMON :

*AMON est un **module** développé par l'équipe EOLE. C'est un **pare-feu** (outil de protection contre les attaques pirates provenant d'Internet) muni d'un **DNS** (Domain Name Server, sert pour la résolution de nom : Nom de la machine ⟺ adresse IP) et d'un **Proxy** (cf. page 17). AMON 2 est la 2ème version de ce module et regroupera de **nouvelles fonctionnalités** dont la QoS.*

2) Introduction à la Qualité de Services (QOS)

a. Pourquoi la QoS ? :

Au début de l'Internet, la préoccupation majeure était de transmettre les paquets **de données** *(1)* à leur destination. Ensuite, des **mécanismes** liés au protocole TCP/IP *(2)* ont été développés pour faire face aux conséquences de la perte de paquets ou de la saturation du réseau. Mais depuis le début des années 1990, la communauté des fournisseurs de service (ISPs) qui administre l'Internet est confrontée non seulement au problème de **croissance explosive** mais aussi à des aspects de globalisation et stabilité du réseau.

La méthode utilisée jusque-là, consistant à fournir des **réseaux surdimensionnés** ne peut plus s'appliquer indéfiniment. Et l'évolution de l'Internet ne permet pas d'offrir une qualité de service constante, ni de donner des priorités à certains types de trafic. C'est pourquoi, les architectes du réseau, les constructeurs et les fournisseurs de service concentrent aujourd'hui leurs efforts sur la définition et la mise en oeuvre de ce concept : **la qualité de service**, *Quality of Service* : QoS.

La Qualité de Service est généralement assimilée à la définition de **classes différenciées de services.** Mais cela signifie aussi **garantir un service** et pour cela **réserver des ressources**. Pour implémenter ces classes ou garantir des ressources, il faut définir une ou plusieurs **politiques** sur les noeuds du réseau permettant d'implémenter la Qualité de Service demandée, en utilisant divers mécanismes (trafic shaping : gestion du trafic réseau, admission control : priorité des flux, etc.).

Pour pouvoir associer une classe de service et garantir des ressources à un certain trafic il faut pouvoir prédire le **comportement du réseau**. Pour cela, il est nécessaire de s'entendre sur la définition de critères de mesures ou **métriques** afin de pouvoir sélectionner le meilleur chemin, lorsque des chemins multiples existent (routage) et vérifier que les demandes en QoS soient satisfaites.

En définitive, la QoS sert à la **gestion de la bande passante** et englobe tous les mécanismes permettant de différencier les types de trafic, ceux-ci pouvant être classés et administrés différemment à travers le réseau. Mais, avant de parler de Qualité de Service, il convient de s'interroger sur **les critères** qui caractérisent le service dans le réseau permettant de délivrer **un service de bonne qualité**.

(1) Paquet :

*Le paquet est **l'unité d'information** utilisée pour communiquer sur le réseau. Ils sont assemblés pour former des messages complets et contiennent les données à transmettre, des en-têtes, et d'autres informations…*

(2) TCP/IP :

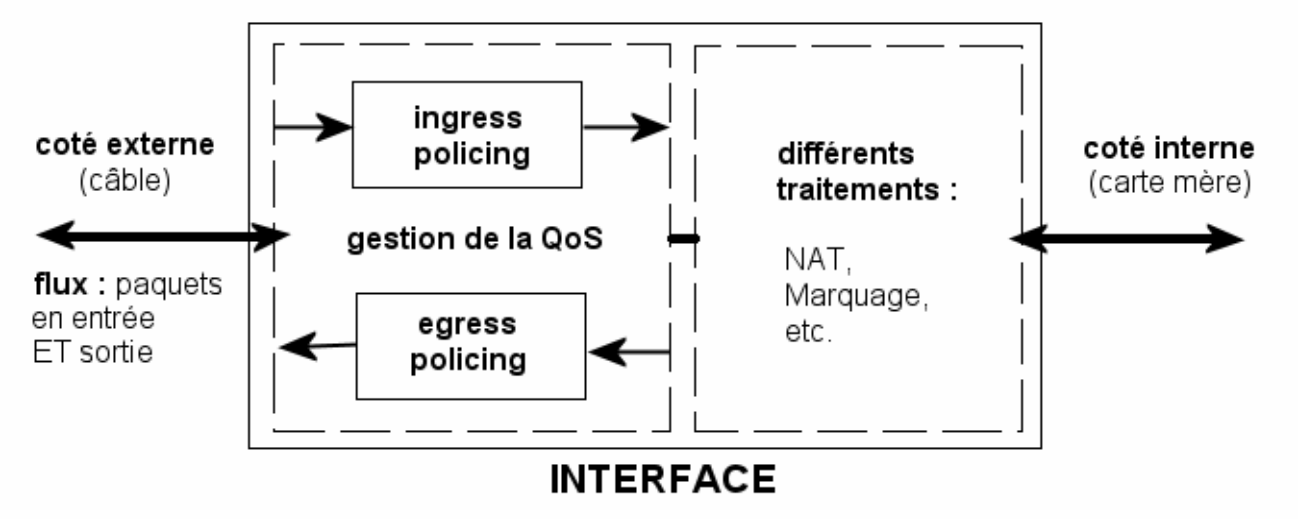
*Le protocole TCP/IP (Transmission Control Protocol/Internet Protocol) est une suite de protocoles. Il représente d'une certaine façon l'ensemble des **règles de communication** sur Internet et se base sur la notion d'adressage IP, c'est-à-dire fournir une adresse IP à chaque machine du réseau afin de pouvoir acheminer des paquets de données.*

b. Caractéristiques de la QoS :

La qualité de service d'un réseau désigne sa capacité à **transporter** dans de bonnes conditions des flux issus de différentes applications. Ces flux doivent être **traités différemment** selon leurs caractéristiques et les objectifs fixés :

- **la fiabilité :** le service d'acheminement des paquets doit être fiable.
- **la bande passante :** suffisante pour absorber les flux générés par les applications.
- **le délai :** rapide pour les applications qui le nécessitent.
- **la régulation :** trafic régulier pour les applications qui le nécessitent.
- **taux d'erreur :** le plus faible possible, garanti aux utilisateurs.

Schéma : Interface supportant la QoS



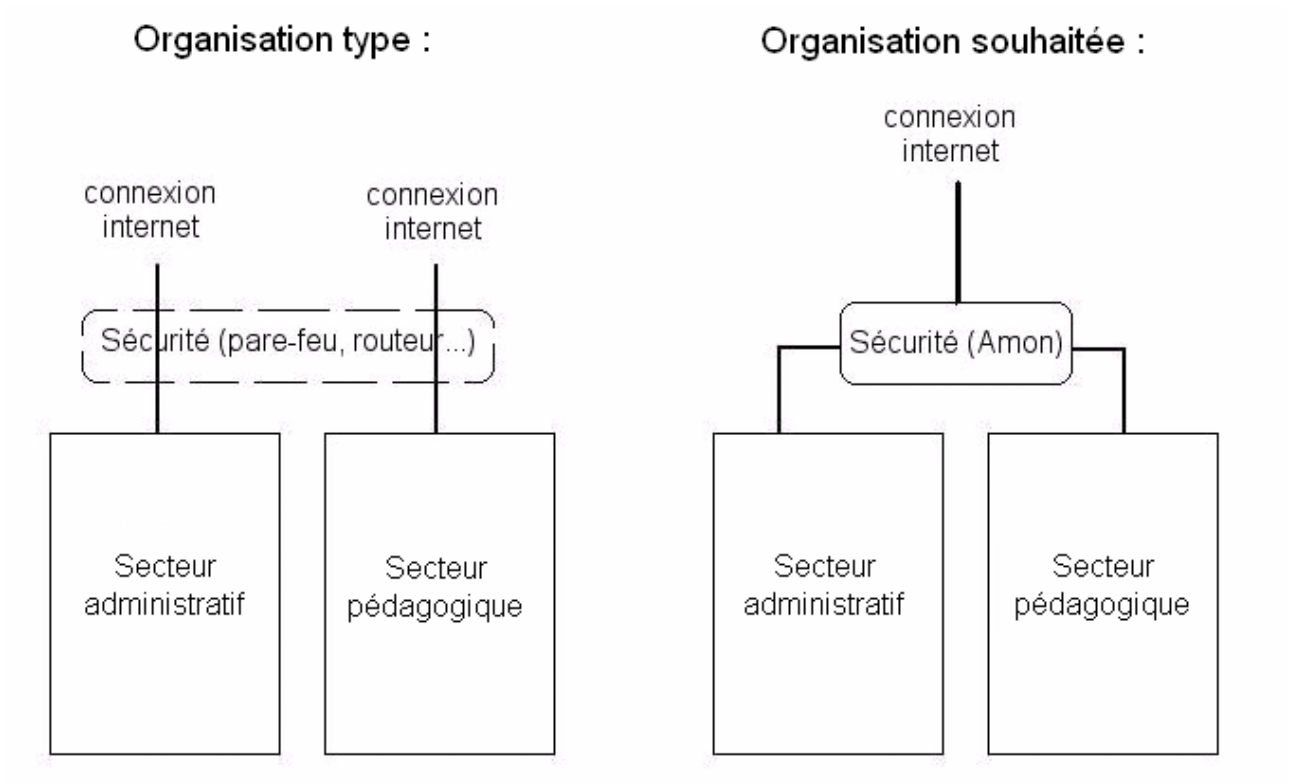
Les paquets arrivent sur l'interface par l'intermédiaire du câble réseau (ou ondes dans le cas du WIFI). Ils passent par la **gestion de la QoS** (ingress : paquets entrants / egress : paquets sortants) puis par les **différents traitements** (NAT, Marquage, etc.) et sortent ensuite de l'interface. Même chose dans l'ordre inverse pour les paquets provenant de la machine (routés : acheminés d'un sous réseau à un autre par une machine appelé routeur).

c. Problématique :

Le protocole TCP/IP n'a pas d'aptitude à connaître les **performances d'un réseau**. Il commence à envoyer des paquets, de plus en plus rapidement et quand des paquets commencent à se perdre, il ralentit. La plupart des files d'attente fonctionnent selon le modèle suivant : elles reçoivent des paquets, les positionnent en file d'attente jusqu'à un certain point, et ensuite, éliminent tous les nouveaux paquets qui arrivent si la file d'attente est pleine. Si on travaille en **UDP**, les paquets ne sont plus retransmis, si c'est du **TCP**, l'émetteur renverra les paquets perdus. Il est donc facile de **monopoliser** la bande passante d'une petite connexion rien qu'en téléchargeant. Nous comprenons donc l'intérêt de la QoS, qui permet (entre autre) de **prioriser** certaines données (ex : administration à distance) afin d'obtenir un **débit constant**.

Du point de vue des **établissements scolaires**, le problème est différent. Le secteur administratif est distinct du secteur pédagogique et il ne faut pas que l'un ou l'autre ne s'approprie toute la bande passante. Il est aussi dommage de posséder une connexion par secteur puisque les connexions actuelles sont amplement suffisantes. C'est ici qu'intervient la QoS qui permet de **partager** la bande passante en 2 parties (une par secteur) afin de garantir à chaque secteur un débit minimum.

Schéma : Organisation type établissement scolaire et organisation souhaitée.



Le projet devant être incorporé à Amon, le système d'exploitation choisi pour mettre en oeuvre la QoS est **Linux** (système d'exploitation utilisé par EOLE).

d. La Qos sous Linux :

La qualité de service est apparue sous Linux avec le **noyau 2.2** et peut être mise en oeuvre grâce à la commande : **tc** *(cf. manuel d'utilisation de la commande : man tc).*

tc qdisc ... : *définition des listes d'attente*
tc class ... : *classification des paquets*
tc filter ... : *filtres sélectifs*

Linux utilise **deux unités de contrôle** du trafic pour la gestion de la bande passante :

- Les **files d'attente** qui décident des flux prioritaires (CBQ, HTB, RED, TBF, SFQ)
- Les **filtres** qui placent les paquets dans les files d'attente (fwmark, u32)

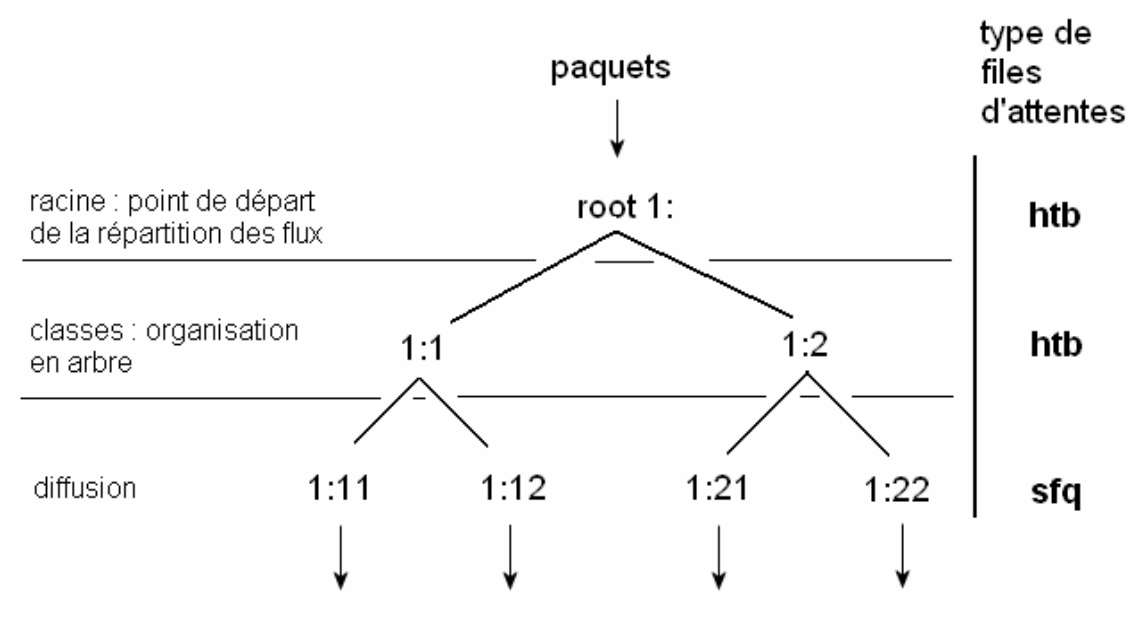
d.1 : Les files d'attente :

Les files d'attente sont gérées grâce à des **gestionnaires** de mise en file d'attente. Il en existe 2 types:

- Les gestionnaires **avec classes** *(qdisc classfull)* : HTB, CBQ et PRIO. Ils sont utilisés pour leur aptitude à gérer les classes, mais ils peuvent également être utilisés sans.
- Les gestionnaires **sans classe** *(qdisc classless)* : fifo, pfifo_fast, red, sfq et tbf. Ils sont utilisés pour gérer les files d'attente lors de la diffusion (envoi des paquets).

L'utilisation des classes n'est pas indispensable mais permet une qualité de services hiérarchisée (donc mieux organisée) et plus stable.

Schéma : Organisation des classes.



Nous allons utiliser le **gestionnaire HTB** *(Hierarchical Token Bucket)* pour définir nos classes, puis le **gestionnaire SFQ** *(Stochastic Fairness Queueing)* pour la gestion des flux. Il existe d'autres gestionnaires mais ils sont plus ou moins performants.

d.2 : Les filtres :

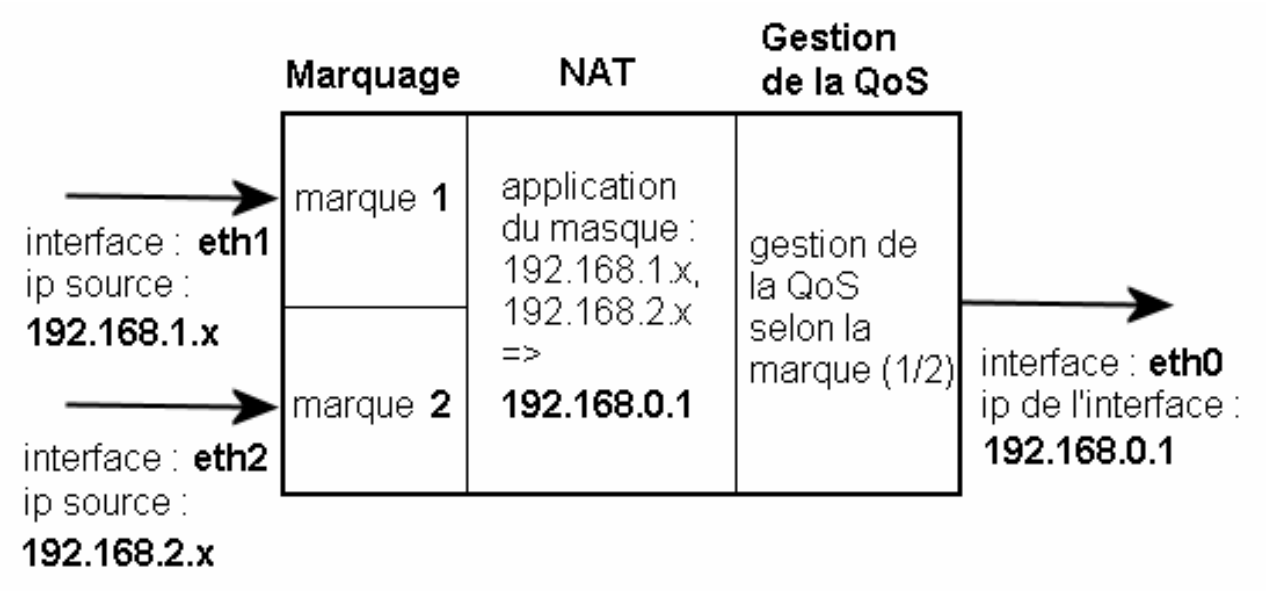
Les filtres permettent d' « aiguiller » les paquets vers les files d'attente. Ils se basent sur les **caractéristiques des paquets** (adresse source, adresse de destination, type de service, marquage, etc.). Il existe plusieurs types de filtres (rsvp, u32, fw, route...) mais nous allons n'en utiliser que 2 :

- **u32**, qui permet de filtrer les paquets en fonction du protocole utilisé, du type de service : ToS, de l'adresse de destination/source, etc. …
- **fw**, qui va dans notre cas, nous permettre de filtrer les paquets en fonction de leurs marques.

Remarques :

- Le pare-feu Amon, sur lequel la QoS va être mise en place, utilise le système de translation d'adresse (NAT) qui s'effectue avant la gestion de la QoS (cf. page suivante). Il n'est donc pas possible de filtrer les paquets sortants par rapport à leur IP source, nous utilisons donc le « marquage de paquets » pour les différencier.

Schéma : Routeur avec gestion de la NAT et de la QoS



Les paquets provenant des **réseaux internes** à l'établissement arrivent sur le routeur sur l'interface eth1 ou eth2 selon leur réseau d'appartenance. Ils sont alors **marqués** pour qu'on puisse les **différencier** : marque 1 pour les paquets provenant du sous réseau 192.168.1.0 et 2 pour le sous réseau 192.168.2.0. Le routeur applique ensuite son « masque » (**la NAT** : translation d'adresse) pour **cacher l'IP source** du paquet, les paquets ont ensuite comme IP source : 192.168.0.1. Les paquets sont alors gérés par la QoS : gestion **en fonction de leur marque** puisque l'IP source du paquet est la même peut importe sa provenance. Enfin, ils sont **acheminés vers leur destination** par le biais de l'interface eth0.

3) Etude de l'existant

Au début du projet, il fallait se familiariser avec la signification de « Qualité de service » et sa mise en oeuvre. Je me suis donc appuyé sur plusieurs supports :

a. Rapport de F. Pastor sur le script HTB :

Fabrice PASTOR est un ancien stagiaire de l'équipe EOLE qui a travaillé sur la Qualité de Service sous Linux. Il a effectué une **première recherche** sur la QoS, en a montré son utilité et a donné un exemple de sa mise en œuvre à l'aide d'un script appelé HTB.init. Il a effectué **plusieurs tests** grâce à un outil de diagnostic réseau et a illustré les résultats sous formes de graphiques.

b. Comparatifs de scripts QoS :

J'ai fait une étude des 3 principaux scripts utilisés pour la gestion de la QoS (en me basant sur des tests effectués sur Internet) :

CBQ.init : file d'attente CBQ

- convient à de petits débits
- nécessite de connaître la taille moyenne des paquets et la vitesse maximale de la connexion
- utilise le temps d'inactivité de la connexion pour calculer une approximation du débit utilisé.

HTB.init : file d'attente HTB

- convient à des gros débits
- consomme peu de ressources
- ne fait pas d'approximation en ce qui concerne le calcul du débit
- nécessite de connaître le débit maximal de la connexion.

Wondershaper : file d'attente HTB

- maintient une bonne réactivité pour le trafic interactif (ssh, telnet...)
- permet de surfer sans souci lors de gros downloads
- s'assure que l'upload ne défavorise pas le download et inversement.

c. Problème et solution :

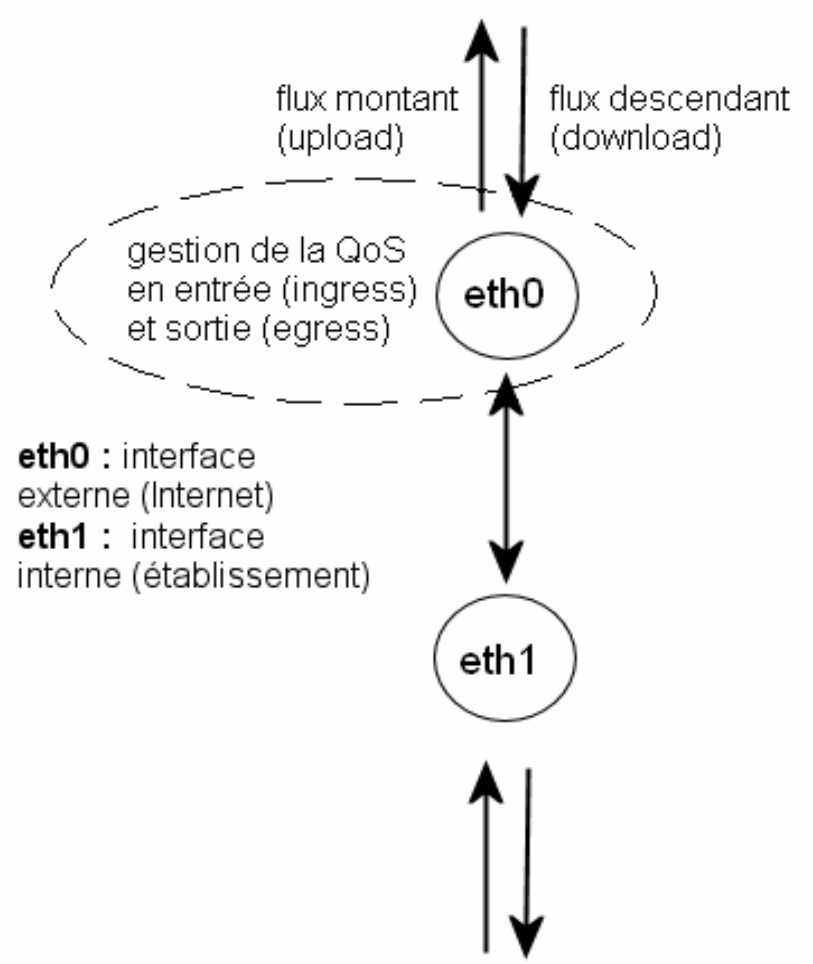
Suite à l'analyse de ces supports, je me suis rendu compte qu'ils **ne répondaient pas aux exigences.** En effet, les scripts proposés étaient **compliqués** à mettre en œuvre (configuration à l'aide de fichiers : un fichier par interface) et étaient **difficilement modifiables**. De plus, aucun script n'était destiné à notre type de **configuration** à savoir la gestion de la QoS sur un pare-feu comportant plusieurs interfaces et muni d'un Proxy (cf page 17) : le module AMON.

J'ai donc opté pour un script **personnalisé** en m'inspirant de ceux existants et en m'aidant de la **documentation** trouvée sur Internet et dans les manuels (man).

4) Déroulement du projet

Pour une approche plus fine du projet, il a fallu procéder par **étapes** :

a. Gestion de 2 interfaces :



Nous allons tout d'abord mettre en œuvre la QoS sur un routeur à **2 interfaces** (eth0 et eth1). Pour ce faire, nous allons créer **un script qui limitera la bande passante** sur l'interface eth0 (flux montant ET descendant).

a.1: Limiter le débit montant (upload) :

tc qdisc add dev eth0 root handle 1: htb default 1

Nous définissons un nouveau gestionnaire de files d'attente (qdisc add) :

- l'interface : dev **eth0**
- l'indice de la « racine » : root handle **1:**
- le gestionnaire de files d'attente : **htb**
- la classe par défaut : default **1**

tc class add dev eth0 parent 1: classid 1:1 htb rate 512kbit

Nous accrochons à la racine une nouvelle classe (class add) :

- sur l'interface : dev **eth0**
- la classe parente : parent **1:** (la racine)
- l'indice de la classe : classid **1:1**
- le gestionnaire de files d'attente : **htb**
- le débit maximum : rate **512kbit**

Remarques :

- Au lieu d'utiliser une classe par défaut, nous aurions pu directement filtrer la « classe » racine.

a.2 : Limiter le débit descendant (download) :

tc qdisc add dev eth0 handle ffff: ingress

Nous définissions :

- l'interface : dev **eth0**
- l'indice de la « racine » : handle **ffff:**
- le type de flux : **ingress** (entrant)

tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip src 0.0.0.0/0 \
police rate 512kbit burst 15k drop flowid :1

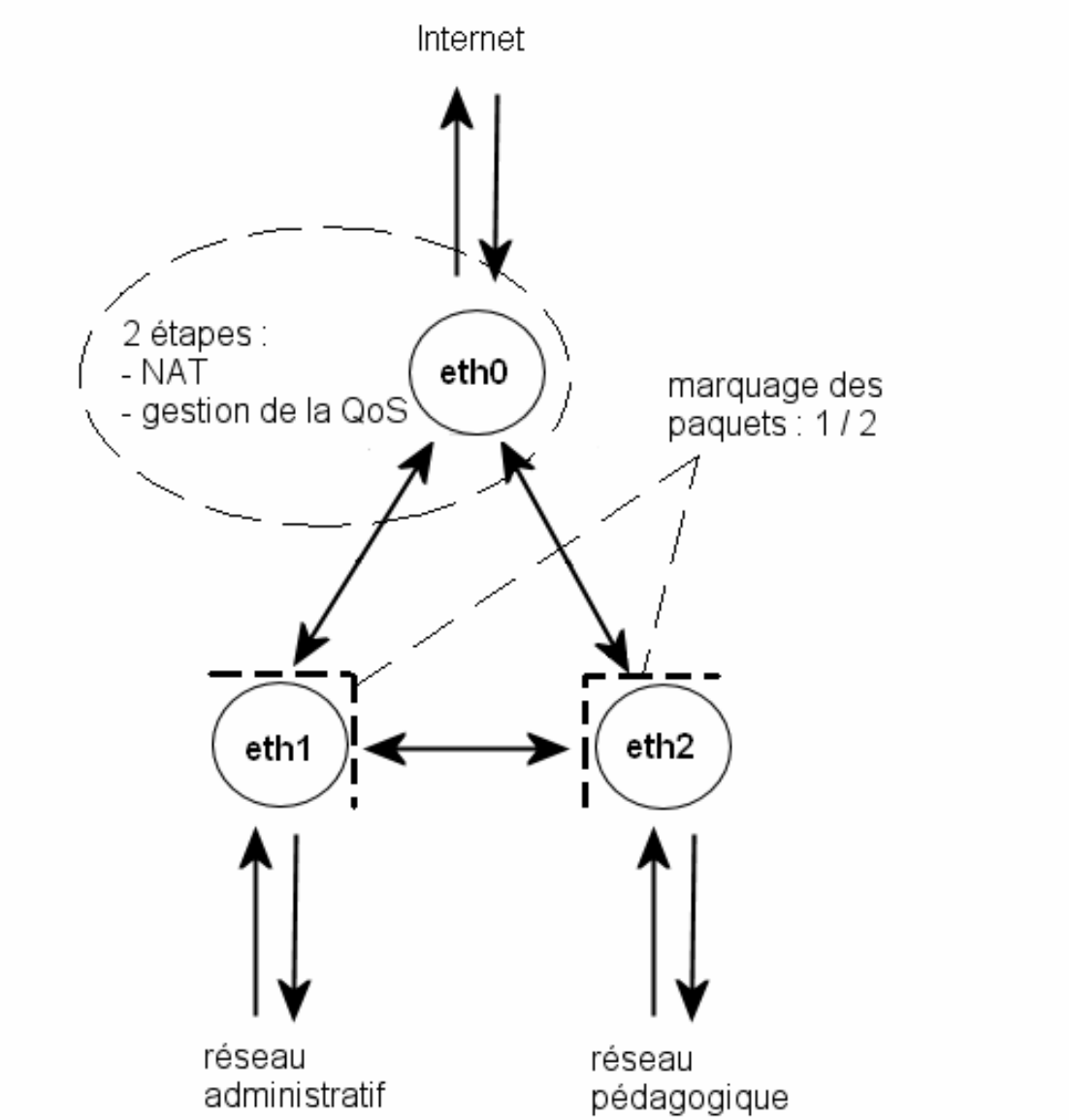
Nous définissons un nouveau filtre appliqué à la racine (filter add) :

- sur l'interface : dev **eth0**
- la classe parente : parent **ffff:** (la racine)
- le protocole utilisé : protocol **ip**
- la priorité : prio **1** (1 = priorité maximum)
- le type de filtre : **u32**
- la source : match ip src **0.0.0.0/0** (tout)
- le débit maximum : police rate **512kbitbit**
- la taille de la file d'attente : burst **15k(bit)**
- le flux sur lequel le filtre est appliqué : drop flowid **:1** (:1 = ffff:)

Remarques:

- l'indice **ffff:** est normalisé et ne concerne que les **flux descendants**.
- pour les flux descendants, il n'y a pas de choix de gestionnaire de files d'attente, les listes sont gérés par le **gestionnaire FIFO** (*first input first output :* premier entré premier sorti). FIFO étant un gestionnaire de liste « classless », on ne peut donc **pas attacher de classes à la racine**, on peut par contre agir directement sur la racine en utilisant des filtres.

b. Gestion de 3 interfaces :



Nous changeons de configuration pour un routeur à **3 interfaces** (eth0, eth1 et eth2). Nous allons toujours travailler sur **l'interface externe** (reliée à Internet) car il n'y a aucun intérêt à limiter la bande passante entre les réseaux de l'établissement (eth1 et eth2) : nous supposons que l'installation réseau de l'établissement est suffisamment récente pour que le débit soit de 100Mb/sec. La bande passante est donc suffisante et il n'y a pas besoin de QoS. Cependant nous nous confrontons au problème posé dans le chapitre 2) d. à savoir la **NAT** (translation d'adresse). Nous allons donc en premier lieu utiliser le **marquage** des paquets pour pouvoir les identifier lors de la gestion de la QoS.

b.1 : Marquage de paquets :

```
iptables -t mangle -A FORWARD -i eth1 -j MARK --set-mark 1
iptables -t mangle -A FORWARD -i eth2 -j MARK --set-mark 2
```

Marquage de paquets avec la commande iptables, nous définissons :

- la table sur laquelle la commande doit opérée : -t **mangle** (plus d'information, se référer au manuel *man iptables*).
- le moment où la commande va être effectuée : -A **FORWARD** (lors du transfert du paquet).
- l'interface : -i **eth1 / eth2**
- l'action à effectuer : -j **MARK** (marquage)
- la marque « injectée » au paquet : --set-mark **1 / 2**

b. 2 : Limiter le débit montant (upload) :

```
tc qdisc add dev eth0 root handle 1: htb default 1
```

Nous passons sur cette commande décrite auparavant.

```
tc class add dev eth0 parent 1: classid 1:1 htb rate 512kbit ceil 1024kbit
tc class add dev eth0 parent 1: classid 1:2 htb rate 640kbit ceil 1024kbit
```

Création des classes, nous définissons :

- l'interface : dev **eth0**
- la classe parent : parent **1:** (racine)
- l'ID de la classe (son numéro) : classid **1:1 / 1:2**
- le gestionnaire de liste utilisé : **htb**
- le débit nominal : rate **512kbit / 640kbit**
- le plafond (si il reste de la bande passante): ceil **1024kbit**

```
tc filter add dev eth0 parent 1: protocol ip handle 1 fw flowid 1:1 # eth1
tc filter add dev eth0 parent 1: protocol ip handle 2 fw flowid 1:2 # eth2
```

Filtrage sur les classes, nous définissons :

- l'interface : dev **eth0**
- la classe parent : parent **1:** (racine)
- le protocole : protocol **ip**
- la marque (repère sur les paquets) :hadle **1 / 2**
- la classe cible : fw flowid **1:1 / 1:2**

```
tc qdisc add dev eth0 parent 1:1 sfq perturb 10
tc qdisc add dev eth0 parent 1:2 sfq perturb 10
```

Gestion des flux, nous définissons :

- l'interface : dev **eth0**
- la classe cible : parent **1:1 /1:2**
- le type de gestionnaire de liste : **sfq**
- des attributs au gestionnaire : **perturb 10** (meilleur résultat suite à différents tests)

b.3 : Limiter le débit montant (upload):

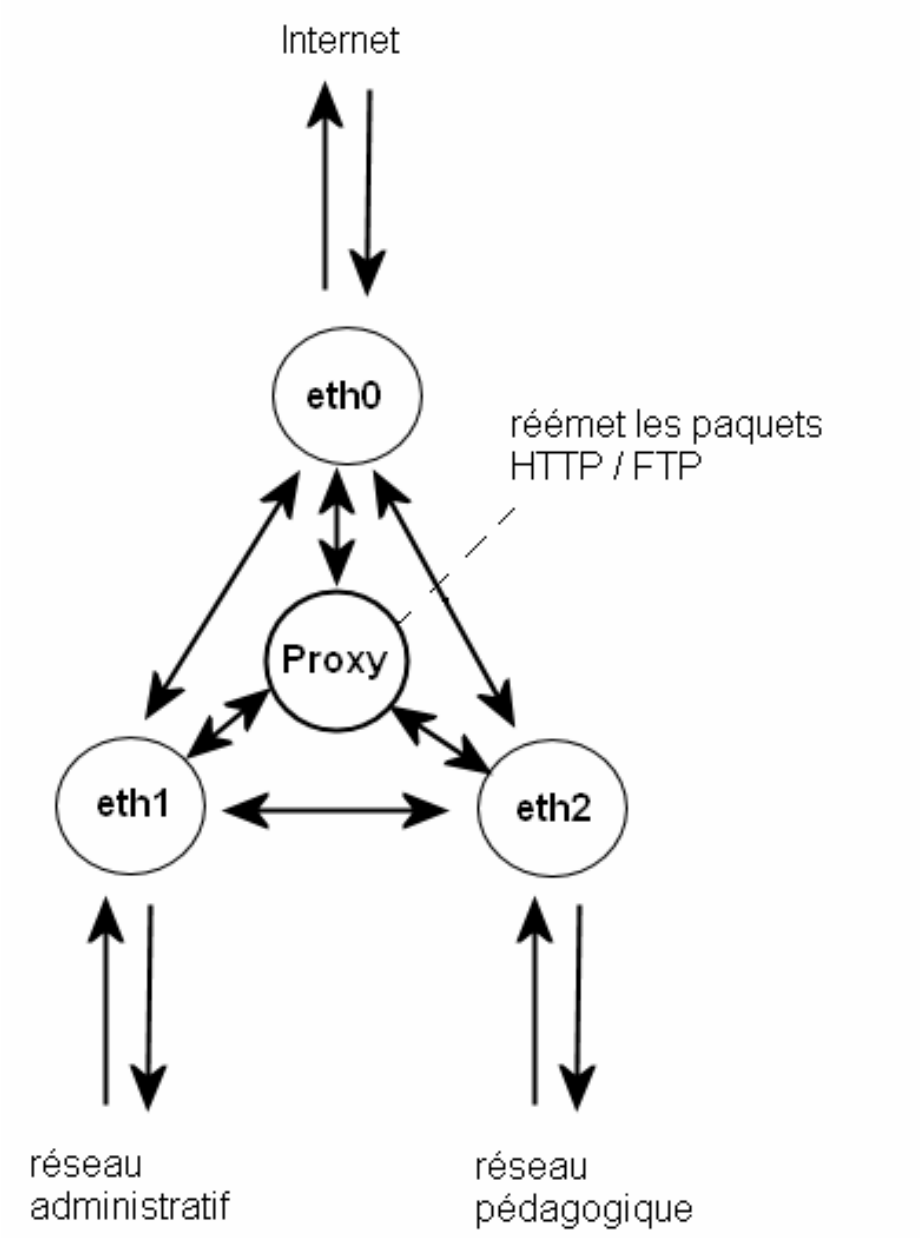
```
tc qdisc add dev eth0 handle ffff: ingress

tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip dst 192.168.1.0/24 \
police rate 128kbit burst 15k drop flowid :1 # eth1
tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip dst 192.168.2.0/24 \
police rate 256kbit burst 15k drop flowid :1 # eth2
```

Gestion des flux, nous définissons :

- l'interface : dev **eth0**
- tout ce qui a été vu auparavant (parent ffff:, etc.)
- la priorité : prio **1 / 1** (soit aucune)
- le filtre utilisé : **u32**
- les attributs du filtre :
  - destination : match ip dst **192.168.1.0/24 / 192.168.2.0/24**
  - le débit : rate **128 / 256kbit**
  - taille des files d'attente : burst **15k(bit)**
  - la cible : drop flowid **:1** (racine)

c. Gestion de 3 interfaces + Proxy + priorité des flux :



Voici la **configuration finale** qui regroupe toutes les contraintes du cahier des charges :

- **routeur à 3 interfaces** :
  - eth0 : interface externe (Internet)
  - eth1 : interface du réseau administratif
  - eth2 : interface du réseau pédagogique
- **gestion de la NAT** par marquage des paquets
- **gestion du Proxy** (cf. page suivante)

Le Proxy réémet les paquets qu'il reçoit. Les paquets ne sont alors plus marqués, il est donc impossible de connaître leur source sur l'interface externe (eth0). Après en avoir parlé au responsable (Luc Bourdot), je lui ai donc proposé une **solution** : créer 3 classes sur l'interface externe (Proxy / pédagogique / administration) en ne leur déléguant qu'une partie de la bande passante.

Chaque classe a donc un débit maximum afin qu'elle ne sature pas toute la bande passante. La classe pour le Proxy se verra attribuer la plus grosse partie de la bande passante du fait qu'elle envoie le plus de paquets (environ 70% de la totalité). Les deux autres classes serviront aux flux de type SMTP (mail), SSH (administration à distance), etc.

c.1 : Rappel sur le fonctionnement du Proxy :

Un serveur Proxy (*serveur mandataire*) est à l'origine une machine faisant fonction d'**intermédiaire** entre les ordinateurs d'un réseau local et internet. La plupart du temps le serveur Proxy est utilisé pour le web, il s'agit alors d'un Proxy HTTP. Toutefois il peut exister des serveurs Proxy pour chaque protocole applicatif (FTP, ...).

Le principe de fonctionnement basique d'un serveur Proxy est assez simple : il s'agit d'un serveur "mandaté" par une application pour effectuer une requête sur Internet à sa place. Ainsi, lorsqu'un utilisateur se connecte à Internet à l'aide d'une application cliente configurée pour utiliser un serveur Proxy, celle-ci va se connecter en premier lieu au serveur Proxy et lui donner sa requête. Le serveur Proxy va alors se connecter au serveur que l'application cliente cherche à joindre et lui transmettre la requête. Le serveur va ensuite donner sa réponse au Proxy, qui va à son tour la transmettre à l'application cliente.

La plupart des Proxys assurent une fonction de **cache**, c'est-à-dire qu'ils gardent en mémoire les pages les plus souvent visitées par les utilisateurs du réseau local afin de pouvoir les leur fournir le plus rapidement possible. Cette fonctionnalité permet d'une part de réduire l'utilisation de la bande passante vers Internet ainsi que de réduire le temps d'accès aux documents pour les utilisateurs.

c.2 : Script QoS :

Marquage des paquets

On utilise iptables afin de différencier les paquets selon leur provenance (à cause de la NAT).

```
iptables -t mangle -A FORWARD -i eth1 -j MARK --set-mark 1
iptables -t mangle -A FORWARD -i eth2 -j MARK --set-mark 2
```

Limitation du débit descendant

On définit les files d'attente, leur débit et les filtres.

```
tc qdisc add dev eth0 root handle 1: htb default 1

tc class add dev eth0 parent 1: classid 1:1 htb rate 1024kbit ceil 2048kbit # Proxy
tc class add dev eth0 parent 1: classid 1:2 htb rate 768kbit ceil 1024kbit # eth1
tc class add dev eth0 parent 1: classid 1:3 htb rate 256kbit ceil 1024kbit # eth2

tc filter add dev eth0 parent 1: protocol ip handle 1 fw flowid 1:2 # eth1
tc filter add dev eth0 parent 1: protocol ip handle 2 fw flowid 1:3 # eth2
```

Définition des flux prioritaires

On filtres aussi les flux prioritaires et on choisit le gestionnaire de file d'attente : sfq.

```
tc filter add dev eth0 parent 1: protocol ip u32 match ip protocol 1 0xff flowid 1:
paquets de type 0x10 (SSH, Telnet...)
tc filter add dev eth0 parent 1: protocol ip u32 match ip tos 0x10 0xff flowid 1:

tc qdisc add dev eth0 parent 1:1 sfq perturb 10
tc qdisc add dev eth0 parent 1:2 sfq perturb 10
tc qdisc add dev eth0 parent 1:3 sfq perturb 10
```

Limitation du débit descendant

On définit la file d'attente racine et les filtres.

```
tc qdisc add dev eth0 handle ffff: ingress

tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip dst 192.168.0.0/24 \
police rate 1024kbit burst 15k drop flowid :1 # Proxy

flux à destination de l'eth1
tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip dst 192.168.1.0/24 \
police rate 128kbit burst 15k drop flowid :1 # eth1

flux à destination de l'eth2
tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip dst 192.168.2.0/24 \
police rate 256kbit burst 15k drop flowid :1 # eth2
```

Définition des flux prioritaires

On filtre les flux prioritaires (ping, ssh, telnet, etc.).

```
tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip protocol 1 0xff flowid :1
tc filter add dev eth0 parent ffff: protocol ip prio 1 \
u32 match ip tos 0x10 0xff flowid :1
```

5) Conclusion

L'engouement pour Internet et les réseaux en général amène à développer de nouvelles techniques pour parer aux limites du protocole TCP/IP. La qualité de services est une de ces techniques et, est en pleine expansion. Elle permet de jouer sur le débit et de prioriser certains flux afin d'obtenir un service garanti aux protocoles et sous réseaux qui le nécessite.
Dans notre cas, la QoS permet d'économiser un abonnement Internet au sein d'un établissement scolaire et de centraliser les connexions du secteur administratif et pédagogique sur un même pare-feu tout en assurant un débit minimum à chaque secteur.

Mon rôle était d'approfondir les recherches effectuées par l'ancien stagiaire (F. Pastor) afin de trouver une solution exploitable sur un Amon. Je pense avoir atteint mes objectifs en proposant un script fonctionnel qui a de plus été testé dans un établissement et reste à ce jour opérationnel (lycée Montchapet à Dijon). Mais mon travail ne peut être incorporé au module car il nécessite un traitement d'automatisation : interface utilisateur, gestion du nombre d'interfaces et des IPs...). Le chef de projet de l'équipe EOLE (mon maître de stage), à la suite de mon départ, a envoyé le script à une équipe annexe située à Toulouse afin de le tester. Si tous les tests se révèlent concluants, l'équipe EOLE reprendra mon travail afin de rendre le script compatible avec le module Amon.